Robust Simulations and Significant Separations

Robust Simulations and Significant Separations

Lance Fortnow[1⋆] and Rahul Santhanam[2⋆⋆]

[1] Northwestern University
[2] University of Edinburgh

Abstract. We define and study a new notion of "robust simulations" between complexity classes which is intermediate between the traditional notions of infinitely-often and almost-everywhere, as well as a corresponding notion of "significant separations". A language L has a robust simulation in a complexity class C if there is a language in C which agrees with L on arbitrarily large polynomial stretches of input lengths. There is a significant separation of L from C if there is no robust simulation of $L \in \mathsf{C}$.

The new notion of simulation is a cleaner and more natural notion of simulation than the infinitely-often notion. We show that various implications in complexity theory such as the collapse of PH if $\mathsf{NP} = \mathsf{P}$ and the Karp-Lipton theorem have analogues for robust simulations. We then use these results to prove that most known separations in complexity theory, such as hierarchy theorems, fixed polynomial circuit lower bounds, time-space tradeoffs, and the recent theorem of Williams, can be strengthened to significant separations, though in each case, an almost everywhere separation is unknown.

Proving our results requires several new ideas, including a completely different proof of the hierarchy theorem for non-deterministic polynomial time than the ones previously known.

1 Introduction

What does the statement "$\mathsf{P} \neq \mathsf{NP}$" really tell us? All is says is that for any polynomial-time algorithm A, A fails to solve SAT on an infinite number of inputs. These hard-to-solve inputs could be exponentially (or much worse) far from each other. Thus even a proof of $\mathsf{P} \neq \mathsf{NP}$ could leave open the possibility that SAT or any other NP-complete problem is still solvable on all inputs encountered in practice. This is unsatisfactory if we consider that one of the main motivations of proving lower bounds is to understand the limitations of algorithms.

Another important motivation for proving lower bounds is that hardness is *algorithmically useful* in the context of cryptography or derandomization. Again, if the hardness only holds for inputs or input lengths that are very far apart, this usefulness is called into question. For this reason, theorists have studied a notion

⋆ Supported in part by NSF grants CCF-0829754 and DMS-0652521.
⋆⋆ Supported in part by EPSRC grant H05068X/1

of almost-everywhere (a.e.) separations, and a corresponding notion of infinitely-often (i.o.) simulations. A language L is in $\mathsf{i.o.C}$ for a complexity class C if there is some $A \in \mathsf{C}$ such that A and L agree on infinitely many input lengths. A class D is almost everywhere not in C if for some language L in D, $L \notin \mathsf{i.o.C}$, that is any C-algorithm fails to solve L on all but a finite number of input lengths. As an example of applying these notions, Impagliazzo and Wigderson [IW97] show that if $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$ then BPP is in $\mathsf{i.o.P}$, and that if $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{o(n)})$, then $\mathsf{BPP} = \mathsf{P}$.

However, the infinitely often notion has its own issues. Ideally, we would like a notion of simulation to capture "easiness" in some non-trivial sense. Unfortunately, many problems that we consider hard have trivial infinitely often simulations. For example, consider any NP-hard problem on graphs or square matrices. The natural representation of inputs for such problems yields non-trivial instances only for input lengths that are perfect squares. In such a case, the problem has a trivial infinitely often simulation on the set of all input lengths which are not perfect squares. On the other hand, the problem could be "padded" so that it remains non-trivial on input lengths which are not perfect squares. It's rather unsatisfactory to have a notion of simulation which is so sensitive to the choice of input representation.

Not unrelated to this point is that analogues of many classical complexity results fail to hold in the infinitely often setting. For example, we do not know if $SAT \in \mathsf{i.o.P}$ implies that the entire Polynomial Hierarchy has simulations infinitely-often in polynomial time. Also it's not true in general that if a complete language for a class is easy infinitely-often, then the entire class is easy infinitely-often. This is true for SAT and NP because SAT is paddable and downward self-reducible, but it's unclear in which situations the implication holds. Given that even these basic analogues are not known, it's not surprising that more involved results such as the Karp-Lipton theorem [KL82] and the theorem of Impagliazzo, Kabanets and Wigderson [IKW02] that $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$ implies $\mathsf{NEXP} = \mathsf{MA}$ don't have known infinitely often analogues either.

In an ideal world, we would like all our algorithms to work on all input lengths, and all our separations to be almost-everywhere separations. While algorithm design does typically focus on algorithms that work on all input lengths, many of the complexity separations we know do not work in the almost everywhere setting. Separations proved using combinatorial or algebraic methods, such as Hastad's lower bound for Parity [Hås86] or Razborov's monotone circuit lower bound for Clique [Raz85] tend to be almost everywhere (in an appropriate input representation). However, such techniques typically have intrinsic limitations, as they run into the natural proofs barrier [RR97]. Many of the lower bounds proved recently have come from the use of indirect diagonalization. A contrary upper bound is assumed and this assumption is used together with various other ideas to derive a contradiction to a hierarchy theorem. These newer results include hierarchy theorems [Bar02,FS04,vMP06], time-space tradeoffs [For00,FLvMV05], and circuit lower bounds [BFT98,Vin05,San07,Wil10a,Wil10b]. Unfortunately, *none* of these results give almost everywhere separations, and so the question

immediately arises what we can say quantitatively about these separations, in terms of the frequency with which they hold.

To address all these issues, we describe a new notion of "robust simulation" and a corresponding notion of "significant separation". A language L is in r.o.C (robustly-often in C) if there is a language A in C such that for every k there are infinitely many m such that A and L agree on all input lengths between m and m^k. A class D has a significant separation from C if there is some L in D such that $L \notin$ r.o.C. This implies that for each $L' \in$ C, there is a constant k such that for each m, L and L' differ on at least one input length between m and m^k. Intuitively, this means that if the separation holds at some input length, there is another input length at most polynomially larger at which the separation also holds, i.e., the hardness is not too "sparsely" distributed.

Our definition of robust simulations extends the notion of uniform hardness of Downey and Fortnow [DF03]. A set A is uniformly hard in the sense of Downey and Fortnow if $A \notin$ r.o.P.

The notion of robust simulation is just slightly stronger than the notion of infinitely often simulation, and correspondingly the notion of significant separation is slightly weaker than that of almost everywhere separations. By making this tradeoff, however, we show that we can often achieve the best of both worlds.

We give robustly often analogues of many classical complexity results, where infinitely often analogues remain open, including

- NP $\subseteq$ r.o.P implies PH $\subseteq$ r.o.P
- NP $\subseteq$ r.o.SIZE($poly$) implies PH $\subseteq$ r.o.SIZE($poly$)
- NEXP $\subseteq$ r.o.SIZE($poly$) implies NEXP $\subseteq$ r.o.MA

We then use these robustly often analogues together with other ideas to give several significant separations where almost everywhere separations remain open, including

- NTIME(n^r) $\not\subseteq$ r.o.NTIME(n^s), when $r > s \geqslant 1$
- For each constant k, Σ_2P $\not\subseteq$ r.o.SIZE(n^k)
- $SAT \not\subseteq$ r.o.DTISP(n^α, polylog(n)) when $\alpha < \sqrt{2}$
- NEXP $\not\subseteq$ r.o.ACC^0

The robustly often notion gives us a cleaner and more powerful theory than the infinitely often notion.

1.1 Intuition and Techniques

To illustrate the advantages of the robustly often notion over the infinitely often notion, let's look at a simple example: trying to show that if SAT is easy, then all of NP is easy. Let $L \in$ NTIME(n^k) be any NP language, where $k > 0$ is a constant. SAT $\in$ i.o.P doesn't immediately imply $L \in$ i.o.P, as the range of the reduction from L to SAT might only intersect input lengths where the polynomial-time algorithm for SAT is incorrect. In this case, the problem can be fixed by padding the reduced instance to polynomially many different input

lengths and using downward self-reducibility to check YES answers for any of these inputs. However, this fix depends on specific properties of SAT.

Showing that $\mathsf{SAT} \in \mathsf{r.o.P}$ implies $L \in \mathsf{r.o.P}$ is an easier, more generic argument. Define a robust set of natural numbers to be any set S such that for each $k > 0$ there is an m for which S contains all numbers between m and m^k for some m. $\mathsf{SAT} \in \mathsf{r.o.P}$ means that there is a robust set S on which the simulation works. Now the reduction from L to SAT creates instances of length $n^k\mathsf{polylog}(n)$, and it's not hard to see that this automatically implies that composing the reduction with the algorithm for SAT gives an algorithm for L which works on some robust subset S' of S. This implies $L \in \mathsf{r.o.P}$. We call a robust subset of a set a *robust refinement*. Many of our arguments will involve defining a series of robust refinements of a robust set such that the desired simulation holds on the final refinement in the series, thus implying that the simulation goes through in the robustly often setting.

Thus far, using robustly often seems easier but hasn't given us any additional power. The situation changes if we consider implications where the assumption is used *two or more* times. An example is the proof that $\mathsf{NP} \subseteq \mathsf{P}$ implies $\mathsf{PH} \subseteq \mathsf{P}$ - this is an inductive proof where the assumption is used several times. Trying to carry an infinitely often simulation through fails miserably in this setting because two infinitely often simulations do not compose - they might work on two completely different infinite sets of input lengths.

Now two robustly often simulations do not in general compose either. It is not in general true that for complexity classes B, C and D, if $\mathsf{B} \subseteq \mathsf{r.o.C}$ and $C \subseteq \mathsf{r.o.D}$, then $\mathsf{B} \subseteq \mathsf{r.o.D}$. However, we *can* get these two robustly often simulations to compose when they are *both* consequences of a single robustly often assumption. The robustly often assumption gives us some robust set to work with. If we're careful we can define a single robust refinement of this set on which $\mathsf{B} \subseteq \mathsf{C}$ holds and so too does $\mathsf{C} \subseteq \mathsf{D}$, which implies $\mathsf{B} \subseteq \mathsf{D}$ holds on this refinement as well.

This is an idea that we will use again and again in our proofs. However, in order to use this idea, we need to be careful with the steps in our proof, as there are only some kinds of implications for which the idea is useful. For example, it works well with fixed polynomial-time reductions or translations with fixed polynomial advice, but not with exponential padding. More importantly, the idea only works when all the steps in the proof follow from a *single* assumption, so we need to re-formulate proofs so that they conform to this pattern. In some cases, eg. the proofs of Theorem 4 and Theorem 6, the re-formulation is non-trivial and leads to proofs that are quite a bit more involved than the originals [IKW02,Kan82].

In the case of hierarchies for non-deterministic time where the lower bound is against robust simulations, the known techniques break down entirely. The traditional argument is a "chaining argument" [Coo72,SFM78,ŽŚ83] which uses a chain of exponentially many input lengths and cannot possibly give a hierarchy against robustly often simulations. Here, we come up with a novel idea of chaining using witnesses to get such a hierarchy for polynomial time.

Our most technically involved result is that the recent breakthrough lower bound of Williams [Wil10a,Wil10b] can be strengthened to a significant separation. The proof of this result uses almost all of the techniques we develop, including the sophisticated use of robust refinements involved in proving Theorem 4, and a variant of the significant hierarchy for non-deterministic polynomial time.

Implicit in our paper is a certain proof system for proving complexity class separations, such that any separation proved in this system automatically yields a significant separation. It's an interesting open problem to make this system explicit, and to study its power and its limitations more formally.

2 Preliminaries

2.1 Complexity Classes, Promise Problems and Advice

We assume a basic familiarity with complexity classes such as P, RP, BPP, NP, MA, AM, Σ_2^p,PP and their exponential-time versions.The Complexity Zoo[3] is an excellent resource for basic definitions and statements of results.

Given a complexity class C, coC is the class of languages L such that $\bar{L} \in \mathsf{C}$. Given a function $s : \mathbb{N} \to \mathbb{N}$, $\mathsf{SIZE}(s)$ is the class of Boolean functions $f = \{f_n\}$ such that for each n, f_n has Boolean circuits of size $O(s(n))$. Given a language L and an integer n, $L_n = L \cap \{0,1\}^n$. Given a class C, $\mathsf{i.o.C}$ is the class of languages L for which there is a language $L' \in \mathsf{C}$ such that $L_n = L'_n$ for infinitely many length n.

In order to deal with promise classes in a general way, we take as fundamental the notion of a complexity measure. A complexity measure CTIME is a mapping which assigns to each pair (M, x), where M is a time-bounded machine (here a time function $t_M(x)$ is implicit) and x an input, one of three values "0" (accept), "1" (reject) and "?" (failure of CTIME promise). We distinguish between *syntactic* and *semantic* complexity measures. Syntactic measures have as their range $\{0,1\}$ while semantic measures may map some machine-input pairs to "?". The complexity measures DTIME and NTIME are syntactic (each halting deterministic or non-deterministic machine either accepts or rejects on each input), while complexity measures such as BPTIME and MATIME are semantic (a probabilistic machine may accept on an input with probability $1/2$, thus failing the bounded-error promise). For syntactic measures, any halting machine defines a language, while for semantic measures, only a subset of halting machines define languages.

A promise problem is a pair (Y, N), where $Y, N \subseteq \{0,1\}^*$ and $Y \cap N = \varnothing$. We say that a promise problem (Y, N) belongs to a class $\mathsf{CTIME}(t)$ if there is a machine M halting in time t on all inputs of length n such that M fulfils the CTIME promise on inputs in $Y \cup N$, accepting on inputs in Y and rejecting on inputs in N.

[3] `http://qwiki.caltech.edu/wiki/ComplexityZoo`

A language L is in $\mathsf{CTIME}(t)/a$ if there is a machine M halting in time $t(\cdot)$ taking an auxiliary *advice* string of length $a(\cdot)$ such that for each n, there is some advice string $b_n, |b_n| = a(n)$ such that M fulfils the CTIME promise for each input x with advice string b_n and accepts x iff $x \in L$. Note that this is a weaker requirement than in the original Karp-Lipton notion where the promise must be satisfied on all advice strings.

For syntactic classes, a lower bound for the class with small advice or for the promise version of the class translates to a lower bound for the class itself. For eg., if there is a promise problem in P which doesn't have polynomial-size circuits, then $\mathsf{P} \not\subseteq \mathsf{SIZE}(poly)$ and similarly, if $\mathsf{P}/O(n) \not\subseteq \mathsf{SIZE}(poly)$, then $\mathsf{P} \not\subseteq \mathsf{SIZE}(poly)$.

Definition 1. *Let S be a subset of positive integers. S is robust if for each positive integer k, there is a positive integer $m \geqslant 2$ such that $n \in S$ for all $m \leqslant n \leqslant m^k$.*

Note that any robust set is infinite. We now define what it means to simulate a language in a complexity class on a subset of the positive integers.

Definition 2. *Let L be a language, C a complexity class, and S a subset of the positive integers. We say $L \in \mathsf{C}$ on S if there is a language $L' \in \mathsf{C}$ such that $L_n = L'_n$ for any $n \in S$.*

Using the terminology of Definition 2, $L \in \mathsf{i.o.C}$ for a language L and complexity class C if there is some infinite set $S \subseteq \mathbb{N}$ such that $L \in \mathsf{C}$ on S. We now define our main notion of robustly-often simulations.

Definition 3. *Given a language L and complexity class C, $L \in \mathsf{r.o.C}$ if there is a robust S such that $L \in C$ on S. In such a case, we say that there is a robustly-often ($\mathsf{r.o.}$) simulation of L in C. We extend this notion to complexity classes in the obvious way - given complexity classes B and C, $\mathsf{B} \subseteq \mathsf{r.o.C}$ if there for each language $L \in \mathsf{B}$, $L \in \mathsf{r.o.C}$. If $\mathsf{B} \not\subseteq \mathsf{r.o.C}$, we say that there is a significant separation of B from C.*

Clearly $\mathsf{B} \subseteq \mathsf{r.o.C}$ implies $\mathsf{B} \subseteq \mathsf{i.o.C}$. Conversely, $\mathsf{B} \not\subseteq \mathsf{i.o.C}$ gives a very strong separation of B and C, i.e., an almost-everywhere separation, while a significant separation is somewhat weaker but still much more significant than simply a separation of B and C. Intuitively, a significant separation means that input lengths witnessing the separation are at most polynomially far apart.

We now define a sequence of *canonical refinements* for any given set S, which will play an important part in many of our proofs.

Definition 4. *Let S be a robust set. The canonical refinement S_d of S at level d is defined as follows for any integer $d > 0$: $m \in S_d$ iff $m \in S$ and $n \in S$ for all $m \leqslant n \leqslant m^d$.*

It is easy to see S_d is robust if S is robust and that $S_d \subseteq S_{d'}$ for $d \geqslant d'$.

Due to space constraints, we omit most proofs in this version of the paper.

3 Robust Simulations

For any NP-complete language L the language

$$L' = \{x10^i \mid x \in L, |x| + 1 + i \text{ is even}\}$$

remains NP-complete but sits in i.o.P. In contrast if any NP-complete set under honest m-reductions sits in r.o.P then NP $\subseteq$ r.o.P.

Lemma 1. *Let L and L' be languages such that L' reduces to L via an honest polynomial-time m-reduction. Let* C *be a complexity class closed under poly-time m-reductions. If there is a robust S such that $L \in$ C on S, then there is a robust refinement S' of S such that $L' \in$ C on S'.*

The proof ideas of Lemma 1 can be used to show that robustly often analogues of various useful implications hold. The first analogue essentially says that we can take a complete language to be representative of a complexity class, even in the context of robustly often simulations. It is an immediate consequence of Lemma 1.

Proposition 1. *If* SAT $\in$ r.o.P*, then* NP $\subseteq$ r.o.P*.*

The next proposition says that translation arguments using a fixed polynomial amount of padding carry through in the robustly often setting, for any "reasonable" complexity measure.

Proposition 2. *Let* BTIME *and* CTIME *be any complexity measures closed under efficient deterministic transductions. Let g and h be time-constructable functions, and p a polynomial. If* BTIME$(g(n)) \subseteq$ r.o.CTIME$(h(n))$*, then* BTIME$(g(p(n))) \subseteq$ r.o.CTIME$(h(p(n)))$*.*

As a consequence of Proposition 2, we get for example that if NTIME$(n) \subseteq$ r.o.P, then NP $\subseteq$ r.o.P.

The proposition below says that simulations of a syntactic class in another class can be translated to a simulation with fixed polynomial advice, even in the robustly often setting.

Proposition 3. *Let* BTIME *be a syntactic complexity measure and* CTIME *a complexity measure, such that both* BTIME *and* CTIME *are closed under efficient deterministic transductions. Let f and g be time-constructable measures and p a polynomial. If* BTIME$(f(n)) \subseteq$ r.o.CTIME$(g(n))$*, then* BTIME$(f(n))/p(n) \subseteq$ r.o.CTIME$(g(n + p(n)))/p(n)$*.*

Theorem 1. *If* NP $\subseteq$ r.o.P*, then* PH $\subseteq$ r.o.P

The proof is by induction, where the inductive hypothesis states that the k'th level of PH is contained in a suitably chosen refinement of the robust set on which the original polynomial-time simulation of SAT works.

Theorem 2. *If* $\mathsf{NP} \subseteq \mathsf{r.o.SIZE}(poly)$*, then* $\mathsf{PH} \subseteq \mathsf{r.o.SIZE}(poly)$

Theorem 3. *If* $\mathsf{NP} \subseteq \mathsf{r.o.BPP}$*, then* $\mathsf{PH} \subseteq \mathsf{r.o.BPP}$.

We omit the proofs of Theorems 2 and 3, which closely resemble the proof of Theorem 1.

Next we state a robust analogue of the Karp-Lipton theorem [KL82]. We formulate a stronger statement which will be useful when we show significant fixed-polynomial circuit size lower bounds for Σ_2^p.

Lemma 2. *If there is a constant* k *and a robust set* S *such that* $\mathsf{SAT} \in \mathsf{SIZE}(n^k)$ *on* S*, then there is a robust refinement* S' *of* S *such that* $\Pi_2 SAT \in \Sigma_2 - \mathsf{TIME}(n^{k+1+o(1)})$ *on* S'.

The following is an immediate corollary.

Corollary 1. *If* $\mathsf{NP} \subseteq \mathsf{r.o.SIZE}(poly)$*, then* $\Sigma_2^p \subseteq \mathsf{r.o.}\Pi_2^p$.

The following can be shown using the easy witness method of Kabanets [Kab01,IKW02] and known results on pseudo-random generators [NW94,KvM99].

Lemma 3. *Let* R *be any robust set and let* $k > 1$ *be any constant. Then there is a robust refinement* R' *of* R *such that either* $\mathsf{NE} \subseteq \mathsf{DTIME}(2^{n^{16k^4}})$ *on* R *or* $\mathsf{MATIME}(n^{4k^2}) \subseteq \mathsf{NE}/O(n)$ *on* R'.

Lemma 3 can be used to prove the following robustly-often analogue of the main theorem of Impagliazzo, Kabanets and Wigderson [IKW02].

Theorem 4. $\mathsf{NEXP} \subseteq \mathsf{r.o.SIZE}(poly)$ *iff* $\mathsf{NEXP} \subseteq \mathsf{r.o.MA}$.

4 Significant Separations

4.1 Hierarchies

The proofs of the hierarchies for deterministic time and space actually give almost-everywhere separations and therefore significant separations.

For nondeterministic time the situation is quite different. Cook [Coo72] showed that $\mathsf{NTIME}(n^r) \subsetneq \mathsf{NTIME}(n^s)$ for any reals $r < s$. Seiferas, Fischer and Meyer [SFM78] generalize this result to show that $\mathsf{NTIME}(t_1(n)) \subsetneq \mathsf{NTIME}(t_2(n))$ for $t_1(n+1) = o(t_2(n))$. Zak [Žák83] gives a simpler proof of the same result. All these proofs require building an exponential (or worse) chain of equalities to get a contradiction. Their proofs do not give almost everywhere separations or significant separations. No relativizable proof can give an i.o. hierarchy as Buhrman, Fortnow and Santhanam give a relativized world that $\mathsf{NEXP} \subseteq \mathsf{i.o.NP}$.

In this section we give a relativizing proof that $\mathsf{NTIME}(n^r) \not\subseteq \mathsf{r.o.NTIME}(n^s)$ for $r > s \geqslant 1$. This also gives a new proof of the traditional nondeterministic time hierarchy.

Theorem 5. *If t_1 and t_2 are time-constructable functions such that*

- $t_1(n) = o(t_2(n))$, *and*
- $n \leqslant t_1(n) \leqslant n^c$ *for some constant* c

then $\mathsf{NTIME}(t_2(n)) \not\subseteq \mathsf{r.o.NTIME}(t_1(n))$.

Corollary 2. *For any reals* $1 \leqslant r < s$, $\mathsf{NTIME}(n^s) \not\subseteq \mathsf{r.o.NTIME}(n^r)$.

Proof (Proof of Theorem 5). Let $M_1, M_2, \ldots$ be an enumeration of multitape nondeterministic machines that run in time $t_1(n)$.

Define a nondeterministic Turing machine M that on input 1^i01^m0w does as follows:

- If $|w| < t_1(i+m+2)$ accept if both $M_i(1^i01^m0w0)$ and $M_i(1^i01^m0w1)$ accept.
- If $|w| \geqslant t_1(i+m+2)$ accept if $M_i(1^i01^m0)$ rejects on the path specified by the bits of w.

Since we can universally simulate $t(n)$-time nondeterministic multitape Turing machines on an $O(t(n))$-time 2-tape nondeterministic Turing machine, $L(M) \in \mathsf{NTIME}(O(t_1(n+1))) \subseteq \mathsf{NTIME}(t_2(n))$. Note $(n+1)^c = O(n^c)$ for any c.

Suppose $\mathsf{NTIME}(t_2(n)) \subseteq \mathsf{r.o.NTIME}(t_1(n))$. Pick a c such that $t_1(n) \ll n^c$. By the definition of r.o. there is some n_0 and a language $L \in \mathsf{NTIME}(t_1(n))$ such that $L(M) = L$ on all inputs of length between n_0 and n_0^c. Fix i such that $L = L(M_i)$. Then $z \in L(M_i) \Leftrightarrow z \in L(M)$ for all $z = 1^i01^{n_0}0w$ for $w \leqslant t_1(i+n_0+2)$.

By induction we have $M_i(1^i01^{n_0}0)$ accepts if $M_i(1^i01^{n_0}0w)$ accepts for all $w \leqslant t_1(i+n_0+2)$. So $M_i(1^i01^{n_0}0)$ accepts if and only $M_i(1^i01^{n_0}0)$ rejects on every computation path, contradicting the definition of nondeterministic time.

4.2 Circuit Lower Bounds

We first state a stronger version of Kannan's [Kan82] lower bound for Σ_2^p against fixed polynomial size, where the separation is significant.

Theorem 6. *For each integer* $k > 1$, $\Sigma_2^p \not\subseteq \mathsf{r.o.SIZE}(n^k)$.

Using similar ideas we can get robustly often analogues of the lower bound of Cai and Sengupta [Cai01] for $\mathsf{S_2P}$ and Vinodchandran [Vin05] for PP:

Theorem 7. *For any* $k > 0$, $\mathsf{S_2P} \not\subseteq \mathsf{r.o.SIZE}(n^k)$.

Theorem 8. *For any* $k > 0$, $\mathsf{PP} \not\subseteq \mathsf{r.o.SIZE}(n^k)$.

Our most technically involved result is that the recent lower bound of Williams [Wil10b] that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$ extends to the robustly often setting. His proof uses the nondeterministic time hierarchy and the proof of Impagliazzo, Kabanets and Wigderson [IKW02], neither of which may hold in the infinitely-often setting. So to get a robustly-often result we require variants of our Theorems 5 and 4. To save space, we will focus on the new ingredients, and abstract out what we need from Williams' paper.

We first need the following simultaneous resource-bounded complexity class.

Definition 5. $\mathsf{NTIMEGUESS}(T(n), g(n))$ *is the class of languages accepted by NTMs running in time* $O(T(n))$ *and using at most* $O(g(n))$ *non-deterministic bits.*

We have the following variant of Theorem 5, which has a similar proof.

Lemma 4. *For any constant* k, $\mathsf{NTIME}(2^n) \not\subseteq \mathsf{r.o.NTIMEGUESS}(2^n/n, n^k)$.

We also need the following robustly often analogue of a theorem of Williams [Wil10a], which uses the proof idea of Theorem 4. The problem SUCCINCT3SAT is complete for NEXP under polynomial-time m-reductions.

Lemma 5. *If* $\mathsf{NE} \subseteq \mathsf{r.o.ACC}^0$ *on* S *for some robust set* S*, then there is a constant* c *and a refinement* S' *of* S *such that* SUCCINCT3SAT *has succinct satisfying assignments that are* ACC^0 *circuits of size* n^c *on* S'.

Proof. The proof of Theorem 4 gives that if $\mathsf{NE} \subseteq \mathsf{ACC}^0$ on S, then there is a constant d and a robust refinement R of S such that SUCCINCT3SAT has succinct satisfying assignments that are circuits of size n^d on R. Since $\mathsf{P} \subseteq \mathsf{ACC}^0$ on S and using Proposition 3, we get that there is a constant c and a robust refinement S' of R such that SUCCINCT3SAT has succinct satisfying assignments that are ACC^0 circuits of size n^c on S'.

Now we are ready to prove the robustly often analogue of Williams' main result [Wil10b].

Theorem 9. $\mathsf{NEXP} \not\subseteq \mathsf{r.o.ACC}^0$.

Proof Sketch. Assume, to the contrary, that SUCCINCT3SAT $\in \mathsf{ACC}^0$ on R for some robust R. By completeness of SUCCINCT3SAT, it follows that there is a robust refinement S of R and a constant $k' > 1$ such that NE has ACC^0 circuits of size $n^{k'}$. Let $L \in \mathsf{NTIME}(2^n)$ but not in $\mathsf{r.o.NTIMEGUESS}(2^n/n, n^k)$, where k will be chosen large enough as a function of k'. Existence of L is guaranteed by Lemma 4. We will show $L \in \mathsf{r.o.NTIMEGUESS}(2^n/n, n^k)$ and obtain a contradiction.

The proof of Theorem 3.2 in Williams' paper gives an algorithm for determinining if $x \in L$. The algorithm non-deterministically guesses and verifies a "small" (size $n^{O(c)}$) ACC^0 circuit which is equivalent to the SUCCINCT3SAT instance to which x reduces, within time $2^n/\omega(n)$ by using Williams' new algorithm for ACC^0-SAT together with the assumption that NEXP and hence P in ACC^0 on S. This guess-and-verification procedure works correctly on some robust refinement of S. Then, the algorithm uses the existence guarantee of Lemma 5 to guess and verify a succinct witness, again using Williams' algorithm for ACC^0-SAT. This further guess-and-verification procedure works correctly on some further robust refinement S'' of S. In total, the algorithm uses at most $n^{dk'}$ non-deterministic bits for some constant d, runs in time at most $2^n/n$ and decides L correctly on S''. By choosing $k > dk'$, we get the desired contradiction. □

Williams' work still leaves open whether $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$. Using the same ideas as in the proof of Theorem 9, we can show that an algorithm for CircuitSAT that improves slightly on brute force search robustly often would suffice to get such a separation.

Theorem 10. *If for each polynomial p,* CircuitSAT *can be solved in time $2^{n-\omega(\log(n))}$ robustly often on instances where the circuit size is at most $p(n)$, then* $\mathsf{NEXP} \not\subseteq \mathsf{SIZE}(poly)$.

4.3 Time-Space Tradeoffs

Proposition 4. *Let t and T be time-constructible functions such that $t = o(T)$. Then* $\mathsf{NTIME}(T) \not\subseteq \mathsf{i.o.coNTIME}(t)$.

Proposition 4 can be used to show the following r.o.analogue of a time-space tradeoff for SAT [FLvMV05].

Theorem 11. *Let $\alpha < \sqrt{2}$ be any constant.* $\mathsf{SAT} \notin \mathsf{r.o.DTISP}(n^\alpha, \mathsf{polylog}(n))$.

References

Bar02. Boaz Barak. A probabilistic-time hierarchy theorem for "Slightly Non-uniform" algorithms. *Lecture Notes in Computer Science*, 2483:194–208, 2002.

BFL91. László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

BFS09. Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against advice. In *Proceedings of 36th International Colloquium on Automata, Languages and Programming*, pages 195–209, 2009.

BFT98. Harry Buhrman, Lance Fortnow, and Thomas Thierauf. Nonrelativizing separations. In *Proceedings of 13th Annual IEEE Conference on Computational Complexity*, pages 8–12, 1998.

Cai01. Jin-Yi Cai. $\mathrm{S}_2^{\mathrm{P}} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, pages 620–629, 2001.

Coo72. Stephen Cook. A hierarchy for nondeterministic time complexity. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 187–192, Denver, Colorado, 1–3 May 1972.

Coo88. Stephen Cook. Short propositional formulas represent nondeterministic computations. *Informations Processing Letters*, 26(5):269–270, 1988.

DF03. Rod Downey and Lance Fortnow. Uniformly hard languages. *Theoretical Computer Science*, 298(2):303 – 315, 2003.

FLvMV05. Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM*, 52(6):833–865, 2005.

For00. L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60(2):337–353, April 2000.

FS04. Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 316–324, 2004.

Hås86. Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

IKW02. Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

IW97. Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997.

Kab01. Valentine Kabanets. Easiness assumptions and hardness tests: Trading time for zero error. *Journal of Computer and System Sciences*, 63(2):236–252, 2001.

Kan82. Ravi Kannan. Circuit-size lower bounds and non-reducibility to sparse sets. *Information and Control*, 55(1):40–56, 1982.

KL82. Richard Karp and Richard Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.

KvM99. Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing*, pages 659–667, 1999.

NW94. Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

Raz85. Alexander Razborov. Lower bounds for the monotone complexity of some boolean functions. *Soviet Mathematics Doklady*, 31:354–357, 1985.

RR97. Alexander Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

San07. Rahul Santhanam. Circuit lower bounds for Merlin-Arthur classes. In *Proceedings of 39th Annual Symposium on Theory of Computing*, pages 275–283, 2007.

SFM78. Joel Seiferas, Michael Fischer, and Albert Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25(1):146–167, January 1978.

Vin05. Variyam Vinodchandran. A note on the circuit complexity of PP. *Theoretical Computer Science*, 347(1-2):415–418, 2005.

vMP06. Dieter van Melkebeek and Konstantin Pervyshev. A generic time hierarchy for semantic models with one bit of advice. In *Proceedings of 21st Annual IEEE Conference on Computational Complexity*, pages 129–144, 2006.

Wil10a. Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 231–240, 2010.

Wil10b. Ryan Williams. Non-uniform ACC circuit lower bounds. Manuscript, 2010.

Žák83. Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, October 1983.